1. Odd String Difference

   Code with output:

```
def f (words):
    def get (word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word)-1)]
    diff = [get (word) for word in words]
    for i in range(len(diff)):
        if diff.count(diff [i]) == 1:
            return words[i]

    return None
words1 = ["adc", "wzy", "abc"]
print(f"Output: {find (words1)}")  # Output: "abc"
words2 = ["aaa", "bob", "ccc", "ddd"]
print(f"Output: {find (words2)}")  # Output: "bob"
```

2. Words Within Two Edits of Dictionary

   Code with output:

```
def t(queries, dictionary):
    def i (word1, word2):
        edits = 0
        for c1, c2 in zip(word1, word2):
            if c1 != c2:
                edits += 1
            if edits > 2:
                return False
        return True

    result = []
    for query in queries:
        for j in dictionary:
            if i (query, j):
                result.append(query)
```

```
            break
    return result
queries1 = ["word", "note", "ants", "wood"]
dictionary1 = ["wood", "joke", "moat"]
print(f"Output: {t(queries1, dictionary1)}")  # Output: ["word", "note",
"wood"]
queries2 = ["yes"]
dictionary2 = ["not"]
print(f"Output: {t(queries2, dictionary2)}")  # Output: []
```

3. Destroy Sequential Targets

Code with output:

```
from collections import defaultdict

def d(nums, space):
    m = d (int)
    min_seeds = {}
    for iin nums:
        m = i% space
        modulo_counts[modulo_class] += 1
        # Track the minimum seed for each modulo class
        if modulo_class not in min_seeds or num <
min_seeds[modulo_class]:
            min_seeds[modulo_class] = num
    max_targets = max(modulo_counts.values())
    best_seed = float('inf')

for modulo_class, count in modulo_counts.items():

        if count == max_targets:
            best_seed = min(best_seed, min_seeds[modulo_class])
return best_seed
nums1 = [3, 7, 8, 1, 1, 5]
space1 = 2
print(f"Output: {destroy_targets(nums1, space1)}")  # Output: 1
nums2 = [1, 3, 5, 2, 4, 6]
space2 = 2
print(f"Output: {destroy_targets(nums2, space2)}")  # Output: 1
```

```
nums3 = [6, 2, 5]
space3 = 100
print(f"Output: {destroy_targets(nums3, space3)}")  # Output: 2
```

4. Next Greater Element IV

Code with output:

```
def f (nums):
    n = len(nums)
    stack = []
    s = {}

    for i in range(n-1, -1, -1):
        while stack and nums[stack[-1]] <= nums[i]:
            stack.pop()

        if stack:
            s[i] = nums[stack[-1]]

        stack.append(i)

    answer = []
    for i in range(n):
        if i in s:
            answer.append(s[i])
        else:
            answer.append(-1)

    return answer
nums1 = [2, 4, 0, 9, 6]
nums2 = [3, 3]

print(f (nums1))  # Output: [9, 6, 6, -1, -1]
print(f(nums2))  # Output: [-1, -1]
```

5.Average Value of Even Numbers That Are Divisible by Three

Code with output:

```
def avg(nums):
    s= 0
    c = 0

    for num in nums:
        if num % 2 == 0 and num % 3 == 0:
            s += num
            c += 1

    if c > 0:
        return s // c
    else:
            return 0
nums1 = [1, 3, 6, 10, 12, 15]
nums2 = [1, 2, 4, 7, 10]

print(avg(nums1))  # Output: 9
print(avg(nums2))  # Output: 0
```

6.Most Popular Video Creator

Code with output:

```
def p(creators, ids, views):
    creator_views = {}
    for i in range(len(creators)):
        creator = creators[i]
        video_id = ids[i]
        view_count = views[i]
        if creator not in creator_views:
            creator_views[creator] = []
        creator_views[creator].append((video_id, view_count))
    max_popularity = -1
```

```
    for creator, videos in creator_views.items():
        total_views = sum(view for _, view in videos)
        max_popularity = max(max_popularity, total_views)
    answer = []
    for creator, videos in creator_views.items():
        total_views = sum(view for _, view in videos)
        if total_views == max_popularity:
            max_views = -1
            most_viewed_video = None
            for video_id, view_count in videos:
                if view_count > max_views or (view_count == max_views and
video_id < most_viewed_video):
                    max_views = view_count
                    most_viewed_video = video_id
            answer.append([creator, most_viewed_video])

    return answer
creators = ["alice", "bob", "alice", "chris"]
ids = ["one", "two", "three", "four"]
views = [5, 10, 5, 4]
print(p(creators, ids, views))  # Output: [["alice", "one"], ["bob", "two"]]
creators = ["alice", "alice", "alice"]
ids = ["a", "b", "c"]
views = [1, 2, 2]
print(p(creators, ids, views))  # Output: [["alice", "b"]]

def digit_sum(num):
```

```
    return sum(int(digit) for digit in str(num))
```

7. Minimum Addition to Make Integer Beautiful

Code with output:

```
def m(n, target):
    if s(n) <= target:
        return 0

    left, right = 0, target + n
    while left <= right:
        mid = (left + right) // 2
        if s(n + mid) <= target:
            right = mid - 1
        else:
            left = mid + 1

    return left
print(m(16, 6))  # Output: 4
print(m(467, 6))  # Output: 33
print(m(1, 1))  # Output: 0
```

8.Split Message Based on Limit

Code with output:

```
def s(message, limit):
    result = []
```

```
    n = len(message)
    start = 0
    c = 1

    while start < n:
        # Determine the length of the current part
        length = min(limit, n - start)
      t = (n + limit - 1) // limit  # This is equivalent to ceil(n / limit)
        part = message[start:start + length] + f"<{c}/{t}>"
        result.append(part)
        start += length
        c += 1

    return result
message1 = "this is really a very awesome message"
limit1 = 9
print(s(message1, limit1))
message2 = "short message"
limit2 = 15
print(s(message2, limit2))
```